**Contents**

[Task 1: Web Server 2](#_Toc155983863)

[1. Open AWS Console. 2](#_Toc155983864)

[2. Open EC2 dashboard using search 2](#_Toc155983865)

[3. Click on Instances. 2](#_Toc155983866)

[4. List of EC2 Instances. 2](#_Toc155983867)

[5. Create an EC2 instance 3](#_Toc155983868)

[6. Secure Shell to Machine 5](#_Toc155983869)

[7. Install LAMP(Linux, Apache, MySQL, PHP) Stack 6](#_Toc155983870)

[8. Verify apache 6](#_Toc155983871)

[9. Configure Apache for 2 Websites 6](#_Toc155983872)

[10. Restart apache 7](#_Toc155983873)

[11. Create Custom website 7](#_Toc155983874)

[12. Check both sites 8](#_Toc155983875)

[13. Open Website to public 8](#_Toc155983876)

[14. Visit Websites 10](#_Toc155983877)

[Task 2: Basic GIT Commands 12](#_Toc155983878)

[1. Introduction 12](#_Toc155983879)

[2. Configuration 12](#_Toc155983880)

[3. Create a Repository 12](#_Toc155983881)

[4. Check Status 15](#_Toc155983882)

[5. Add Files to Staging Area 15](#_Toc155983883)

[6. Commit Changes 15](#_Toc155983884)

[7. View History 18](#_Toc155983885)

[8. View Contents of Older Commit 19](#_Toc155983886)

[9. Restore Previous Commit 20](#_Toc155983887)

[Git Commands 22](#_Toc155983888)

# Task 1: Web Server

Considering you have already created a free-tier AWS account. Follow the following steps:

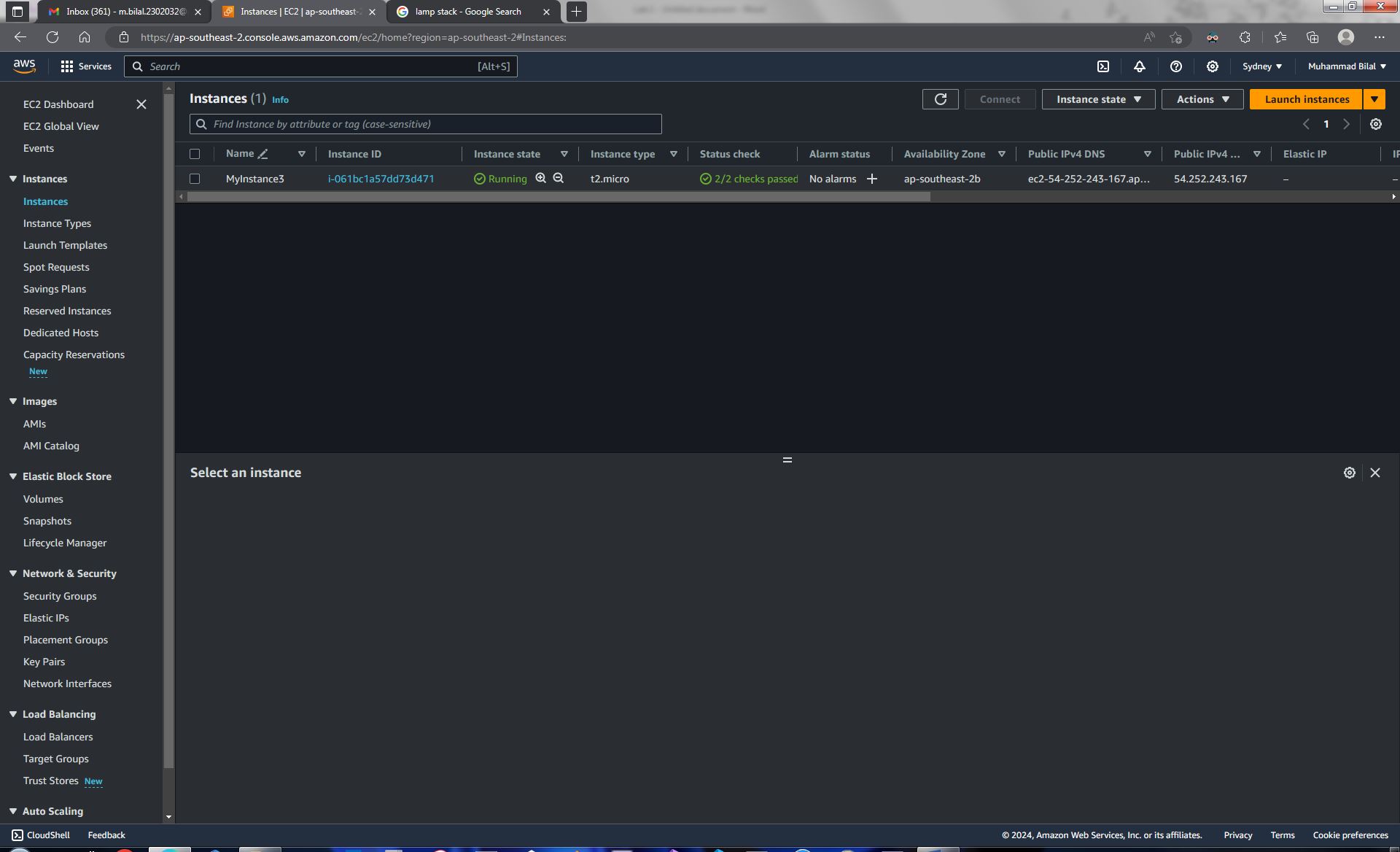
## Open AWS Console.

## Open EC2 dashboard using search

## Click on Instances.

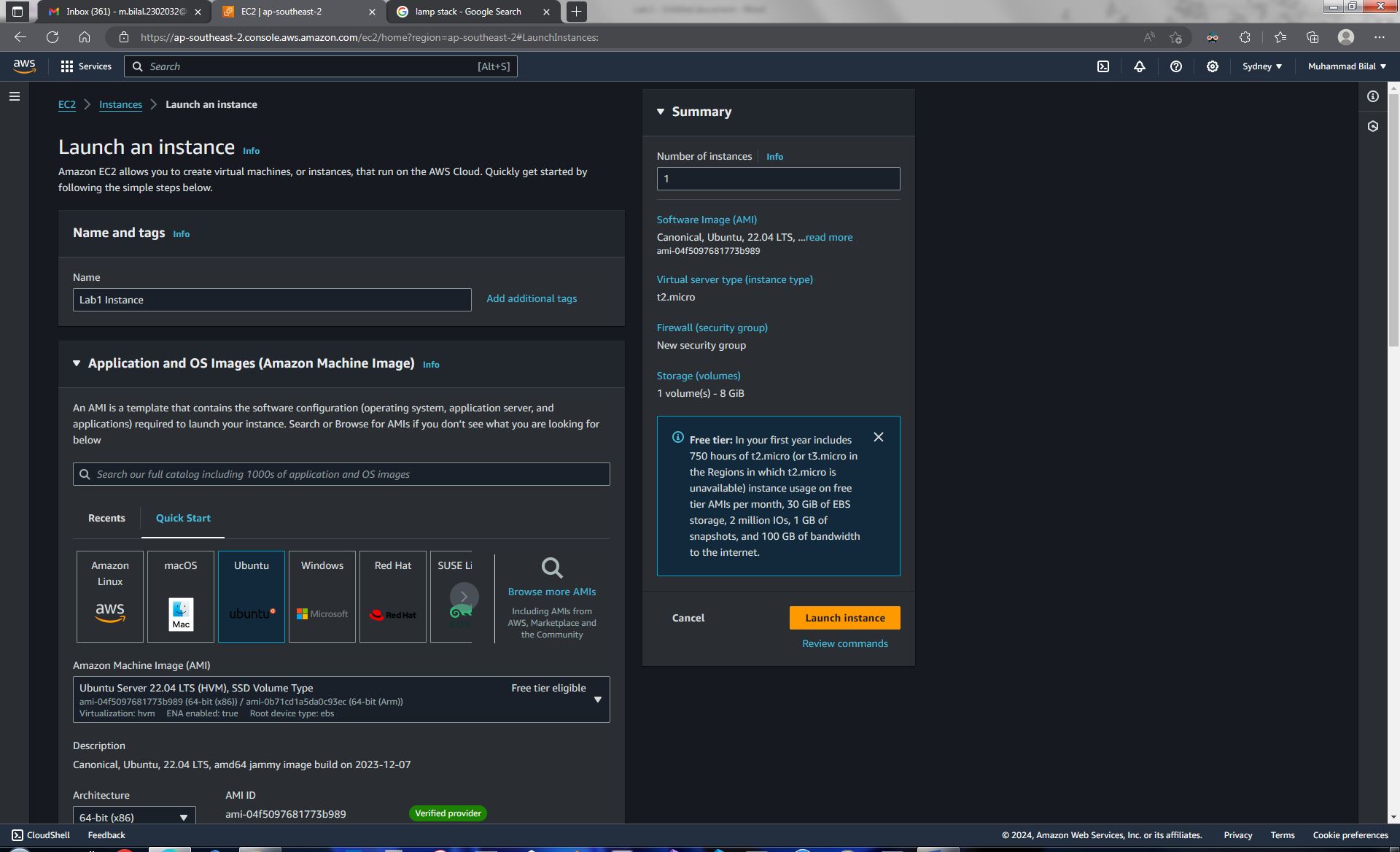
## List of EC2 Instances.

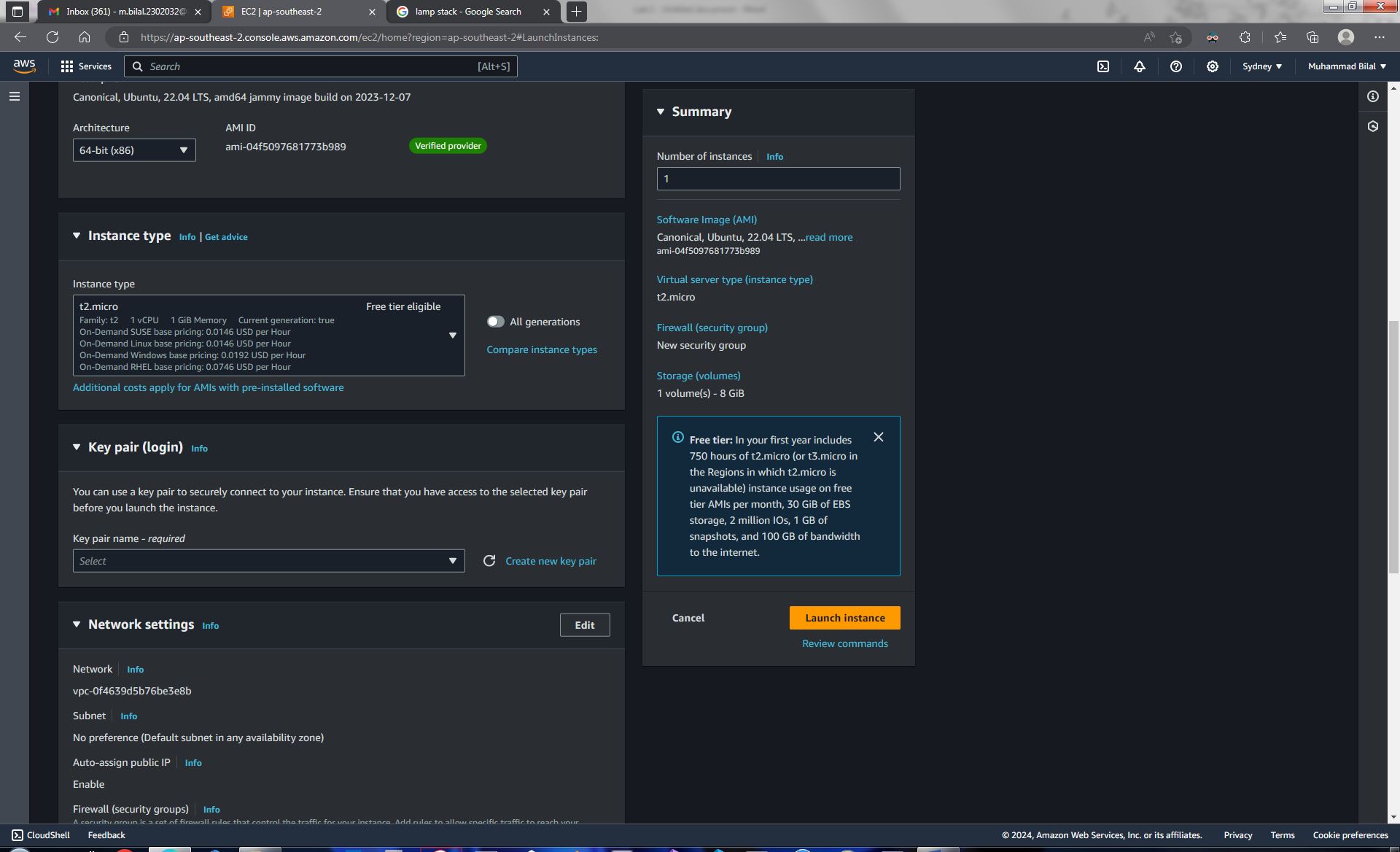
You should be able to see your EC2 instances as shown in the image below.

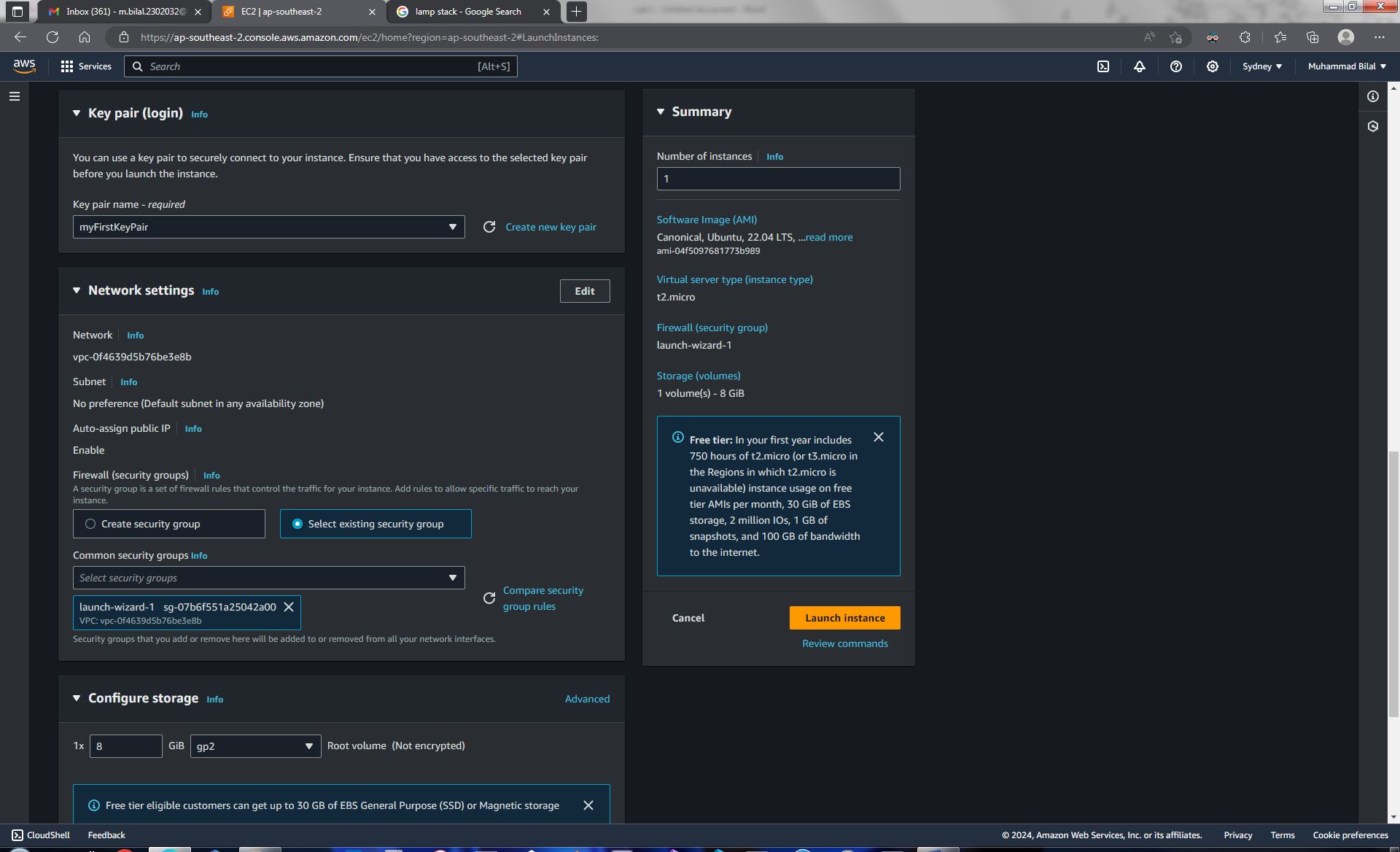


## Create an EC2 instance

Click on launch instance. Then create an EC2 instance with Ubuntu x64 on t2.micro machine. Configuration should be as shown in the below images. Key pair can be of your choice.



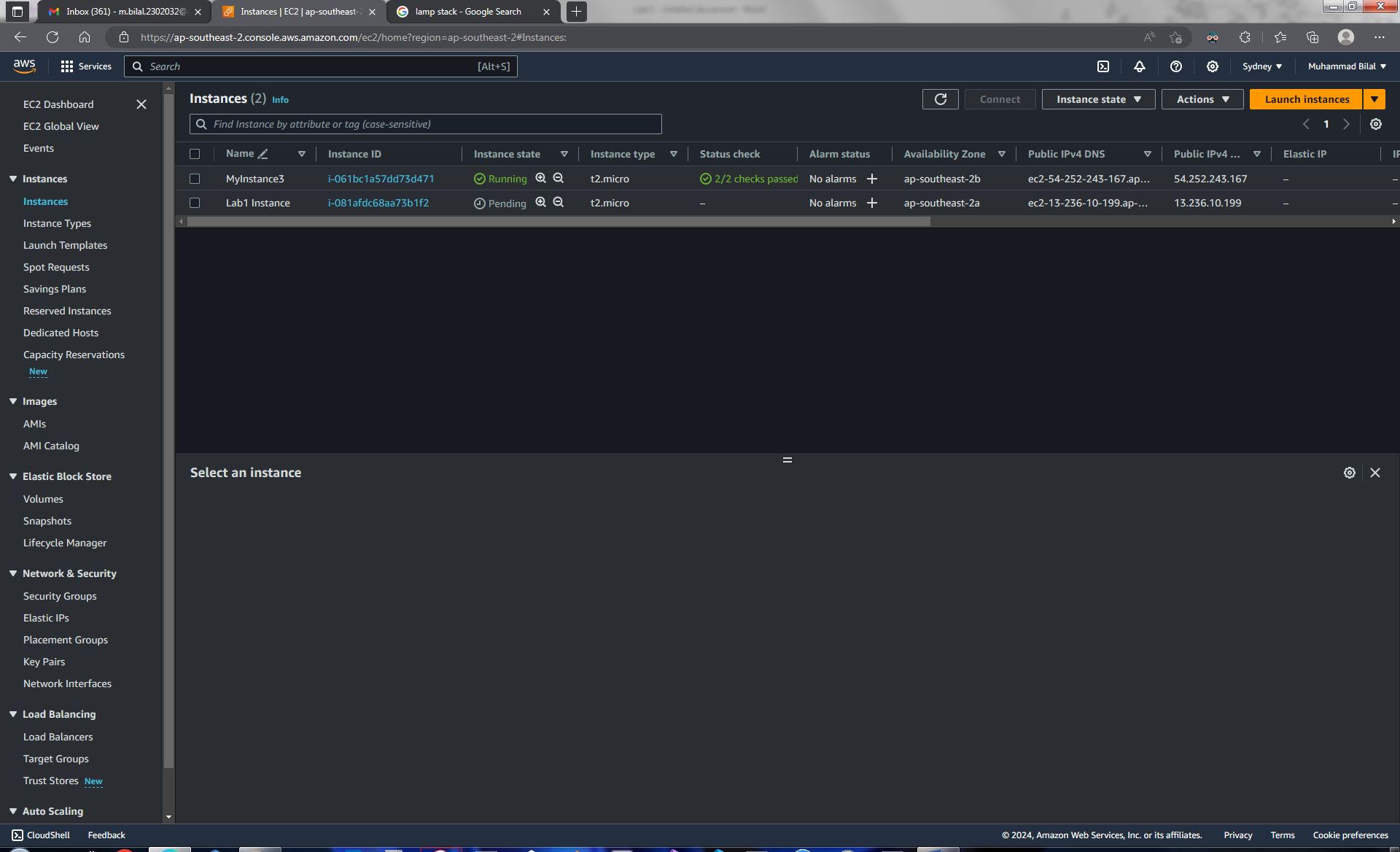




Leave the rest untouched and click on **Launch Instance**.

## Secure Shell to Machine

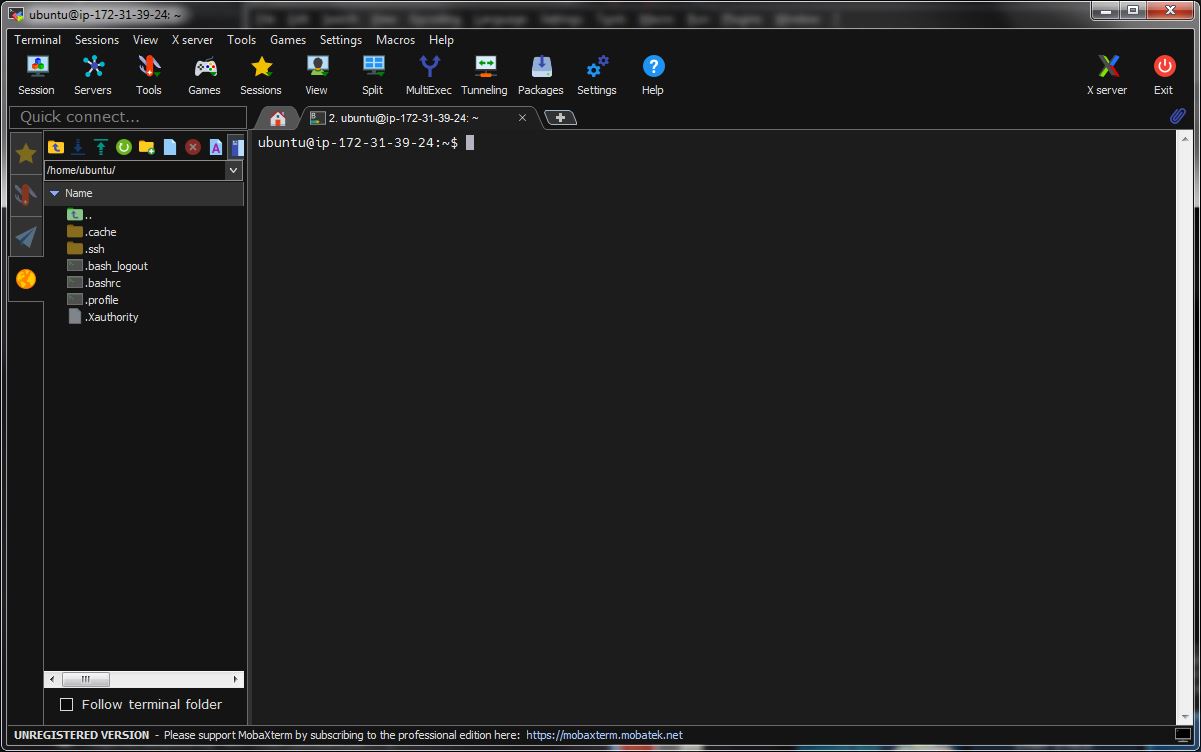
You can view your EC2 instance’s public IPv4 DNS from the list of EC2 instances that you opened before.



Copy your instance’s IPv4 DNS and create an ssh command in a text editor as follows:

ssh -i [KeyPairName] ubuntu@[Instance’s Public IPv4 DNS]

Now open a terminal that supports ssh. Change current directory to where your key pair file is saved. I am using MobaXterm terminal. Copy and paste the above ssh command to terminal. Press enter. You should be connected to your EC2 instance through secure shell. Your terminal should look something like as shown in the below image.



## Install LAMP(Linux, Apache, MySQL, PHP) Stack

Create a shell script file using:

sudo touch nano script.sh

Write the following lines, save the file and exit nano:

sudo apt update

sudo apt install apache2

sudo apt install mysql-server

sudo apt install php

clear

echo "DONE"

Now allow executable permissions to the file script.sh by executing:

sudo chmod +x script.sh

Run script using:

sudo ./script.sh

Enter yes for each prompt during installation. After execution completes, your terminal should show **DONE.**

## Verify apache

Run:

curl localhost

to confirm apache server is running on you instance on port 80. Your terminal should show contents of an html file.

## Configure Apache for 2 Websites

Edit apache.conf by executing:

sudo nano /etc/apache2/apache.conf

Insert Line:

Listen 81

to allow apache to listen on port 80 as well as on 81. Save the file and exit nano. Now, edit sites-available by executing:

sudo nano /etc/apache2/sites-available/site2.conf

Insert following lines to file:

<VirtualHost \*:81>

ServerAdmin admin@admin.com

ServerName localhost

DocumentRoot /var/www/html/site2

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Save file and exit nano.

## Restart apache

To enable sites on apache server, first disable all sites and then enable all sites by executing:

sudo a2dissite (then enter \* for all)

sudo a2ensite (then enter \* for all)

sudo systemctl restart apache2

This will restart apache server with two sites hosted on ports 80 and 81 respectively.

## Create Custom website

Now edit first webpage using nano:

sudo nano /var/www/html/index.html

Create and edit 2nd website using nano:

mkdir /var/www/html/site2

sudo nano /var/www/html/site2/index.html

## Check both sites

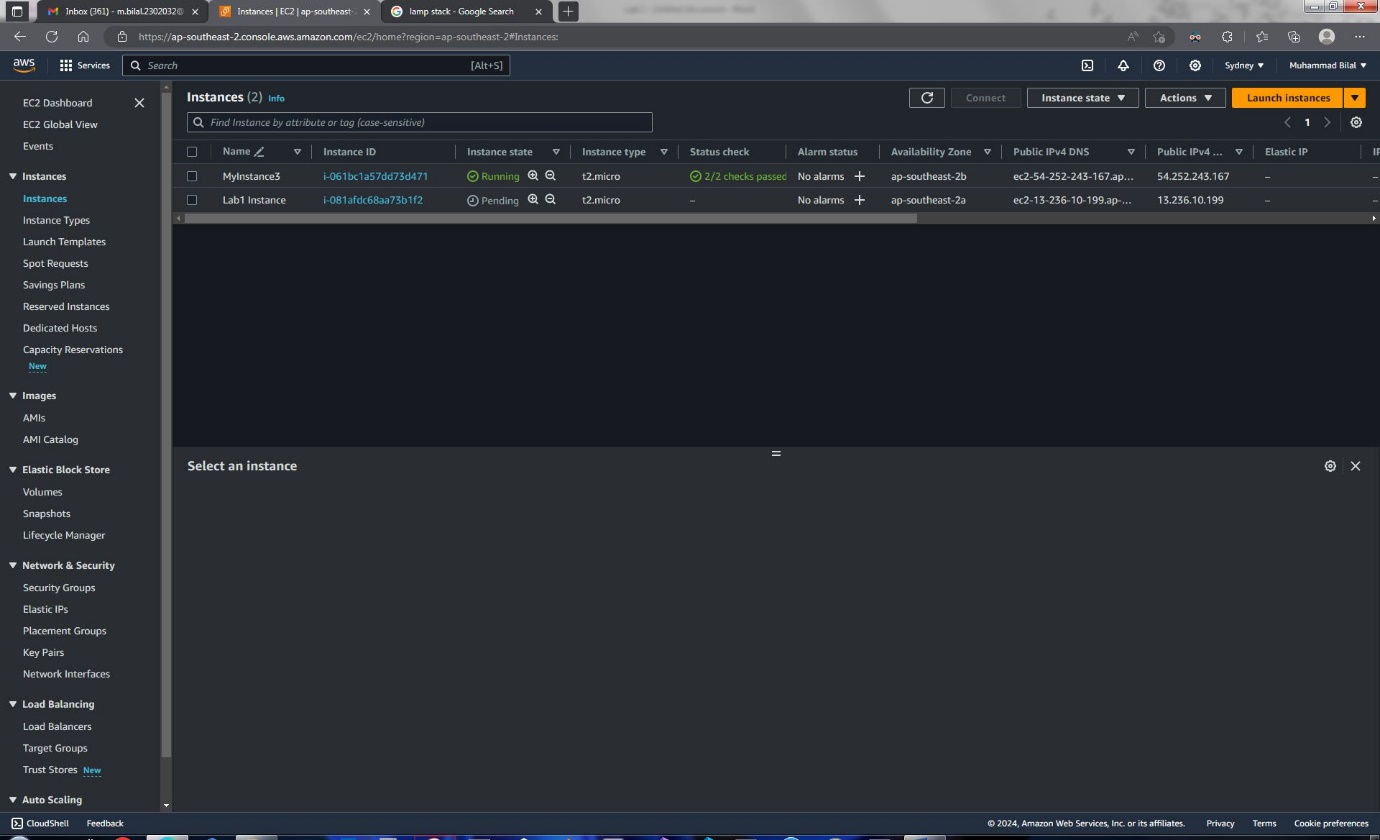
You can check both sites using curl commands. Both curl commands should return the html file contents that you inserted in your two websites.

curl localhost

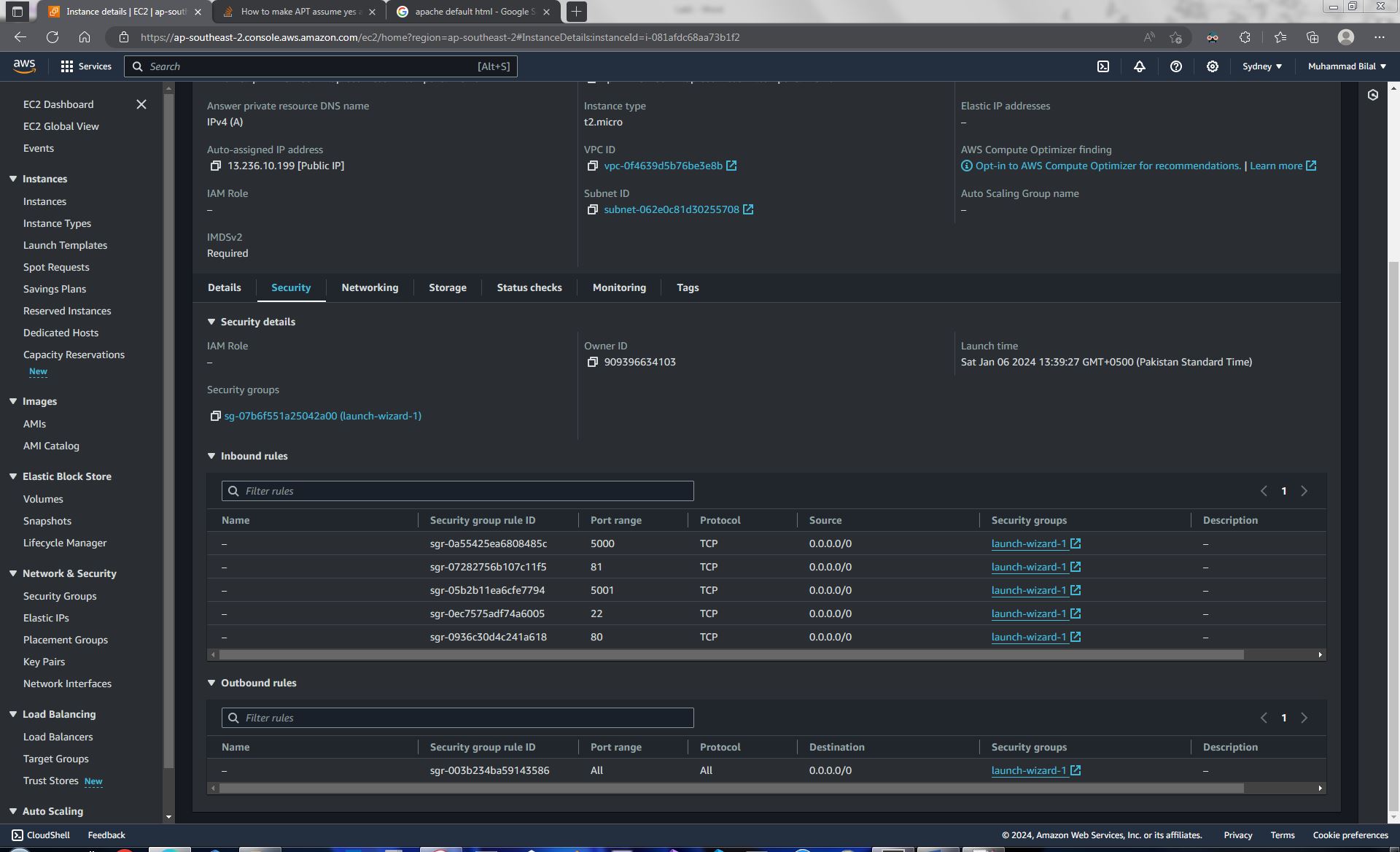
curl localhost:81

## Open Website to public

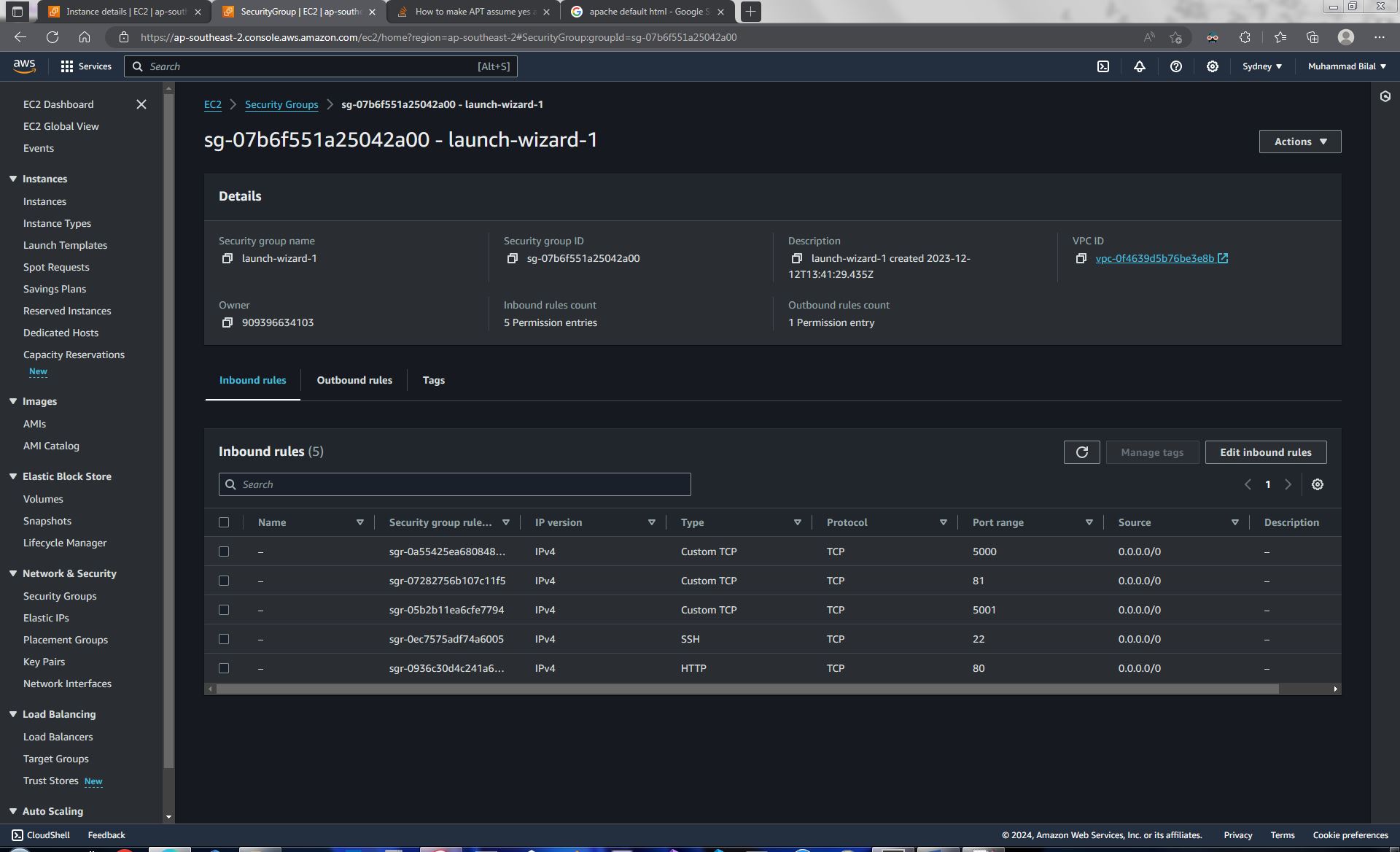
Go to AWS console. Open your EC2 Instance.



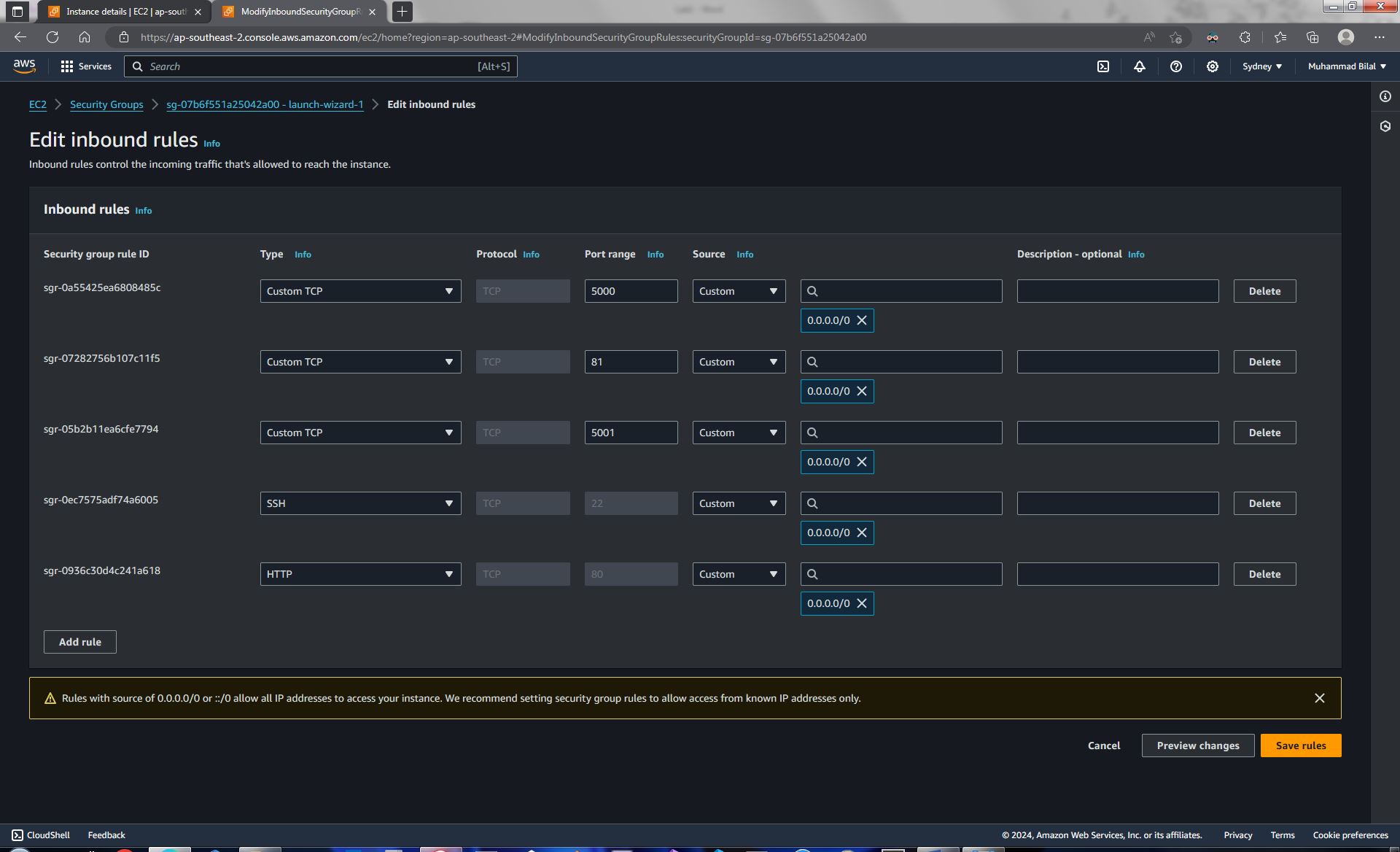
Under Security tab, click on the assiciated security group. This will open your security group page. Click on your security group ID to open it.



Click “Edit Inbound Rules”.



Click Add Rule to add a new rule. Choose TCP protocol with port 80 and source as anywhere IPv4. Perform similar steps to open port 81 to public. Click save rules.

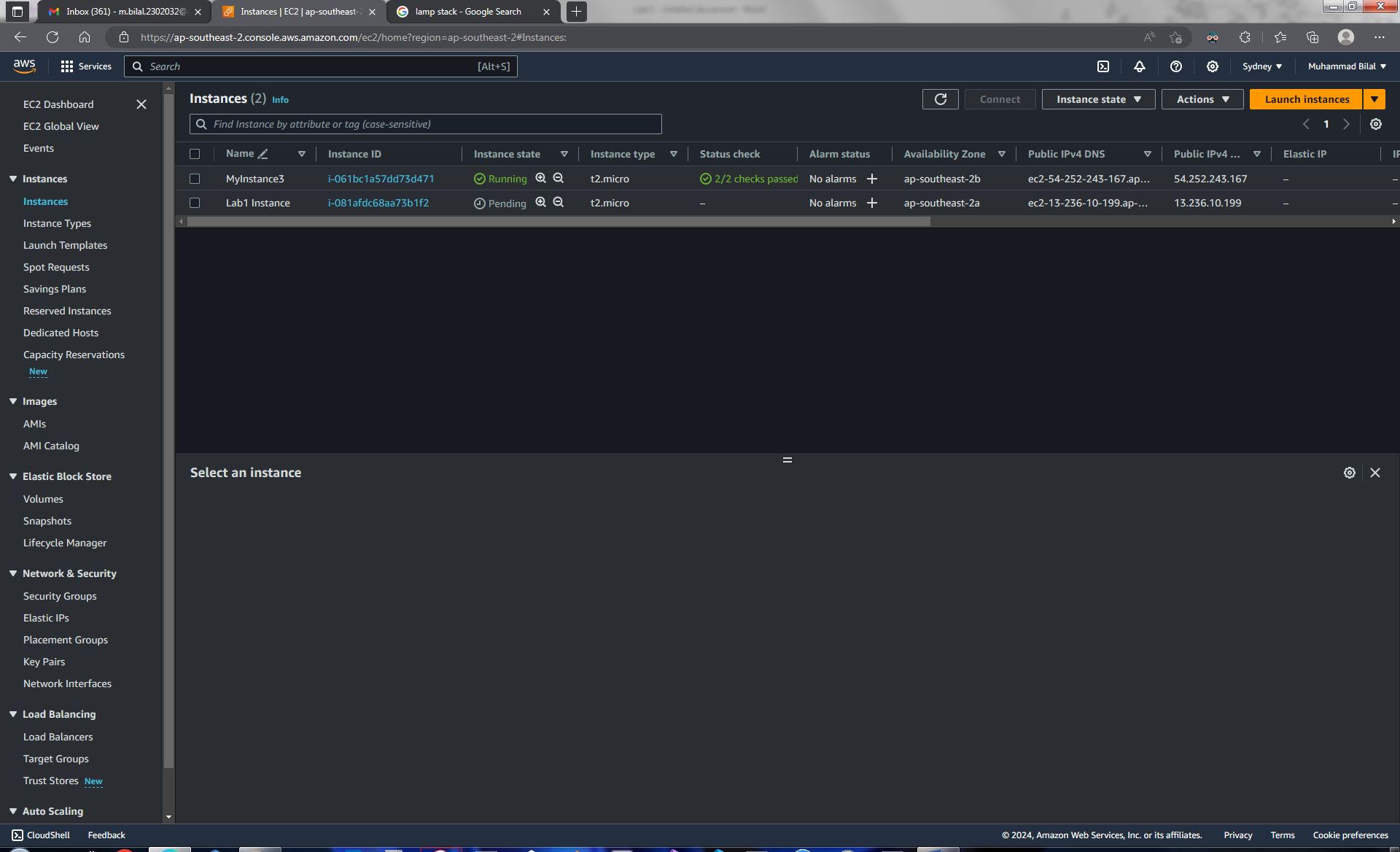


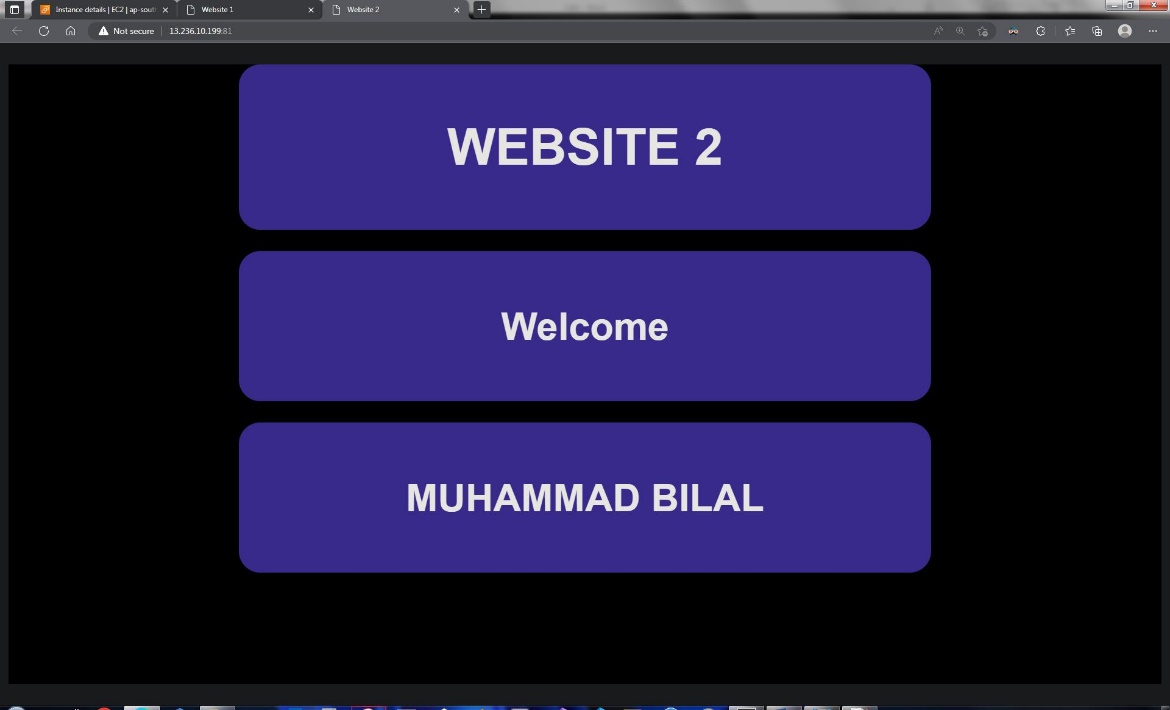
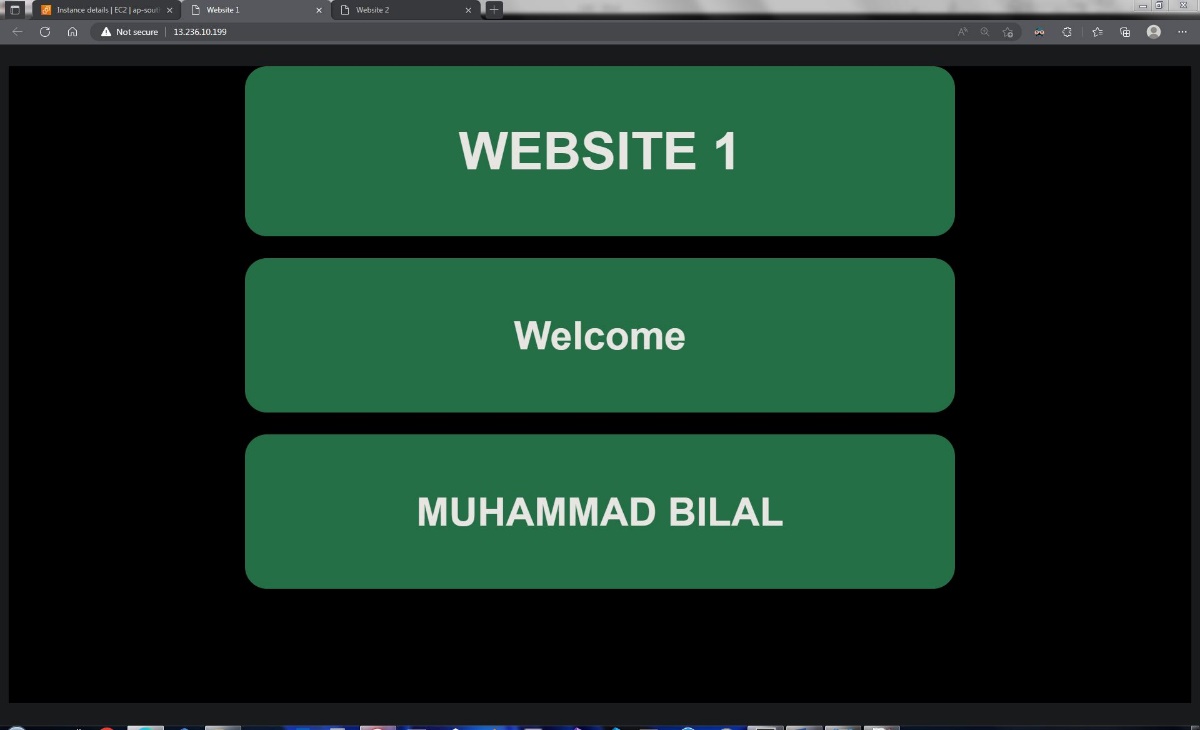
## Visit Websites

You can now open your both websites using your isntance’s public IPv4 address and port number 80 and 81.

Visit in your web browser:

[Instance’s public IPv4 address]:[80 or 81]





# Task 2: Basic GIT Commands

## Introduction

Repository is a place where you can store your code, files and folders. Typically, a single repository contains files for a single project. Git is a distributed version control system that allows users to track repository’s history and to collaborate on repositories remotely. We will now create a Git repository and push to GitHub. Then, we will also see basic concepts of git branching.

## Configuration

First , We need to Configure our Git client. Run:

git config --global user.name "Muhammad Bilal"

git config --global user.email "bilal.asim5555@outlook.com"

git config --global core.editor "code --wait"

git config --global core.autocrlf true

git config --global diff.tool "vscode"

git config --global difftool.vscode.cmd "code --wait --diff $LOCAL $REMOTE"

Above commands configure git’s username, email, default code editor, autocrlf, diff tool respectively. If you are on Windows set autocrlf to true otherwise set it to input. Autocrlf tell git how to manage carriage return and line feeds as new line characters in files. Diff Tool is used for comparing difference between two files visually.

## Create a Repository

Go to location where you want to create a repository.

Run command:

mkdir Lab1

The Lab1 folder would represent our repository. Now, move to created folder by running:

cd Lab1

Create 2 files using vim and store your custom websites in them. Run:

vim index1.html

paste:

<style>

body{

background-color: black;

color: white;

font-family: arial;

font-size: 150%;

}

.box{

background-color: seagreen;

text-align: center;

color: white;

border-radius: 20px;

padding: 20px;

padding-top: 50px;

padding-bottom: 50px;

margin: 20px;

margin-right: 20%;

margin-left: 20%;

}

</style>

<html>

<head>

<title> Website 1 </title>

</head>

<body>

<h1 class="box">WEBSITE 1</h1>

<h2 class="box">Welcome</h2>

<h2 class="box">MUHAMMAD BILAL</h2>

</body>

</html>

Press :**wq** and press enter. Now follow similar steps to save second site in index2.html. Run:

vim index2.html

Paste:

<style>

body{

background-color: black;

color: white;

font-family: arial;

font-size: 150%;

}

.box{

background-color: SlateBlue;

text-align: center;

color: white;

border-radius: 20px;

padding: 20px;

padding-top: 50px;

padding-bottom: 50px;

margin: 20px;

margin-right: 20%;

margin-left:20%;

}

</style>

<html>

<head>

<title> Website 2 </title>

</head>

<body>

<h1 class="box">WEBSITE 2</h1>

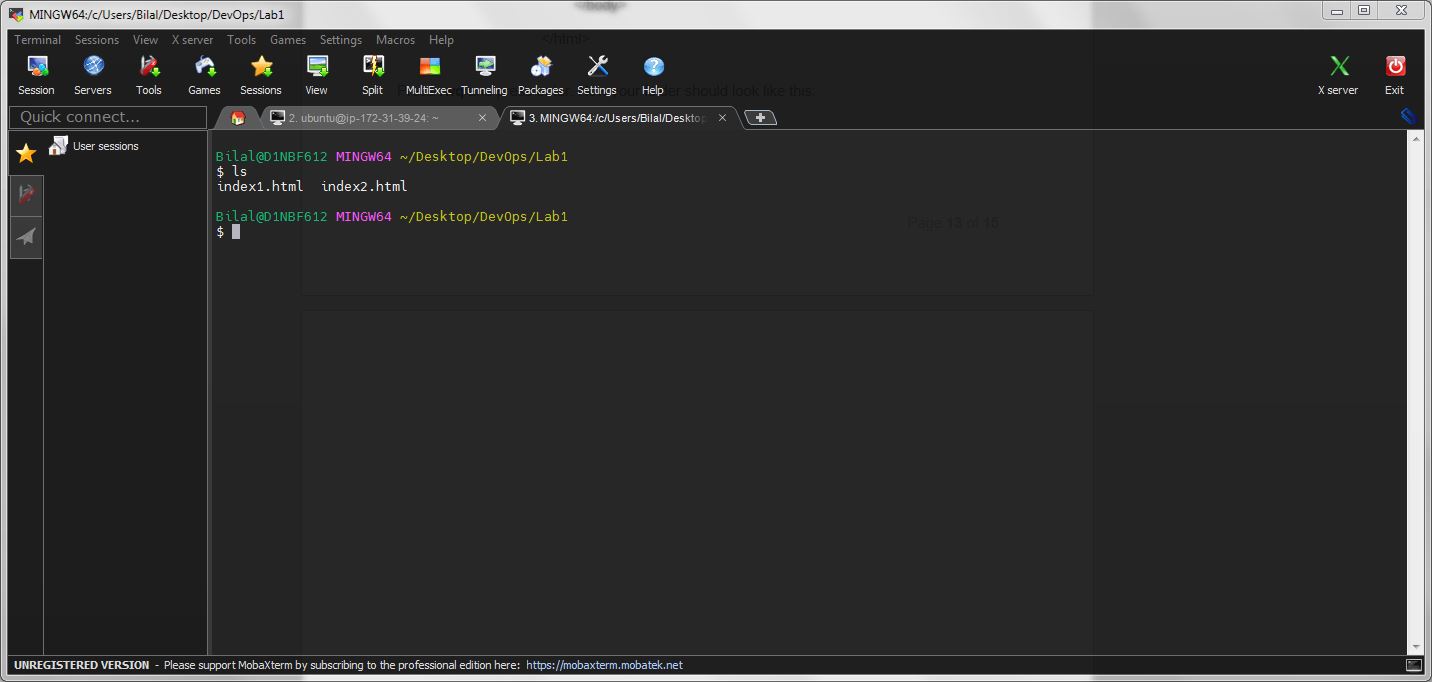
<h2 class="box">Welcome</h2>

<h2 class="box">MUHAMMAD BILAL</h2>

</body>

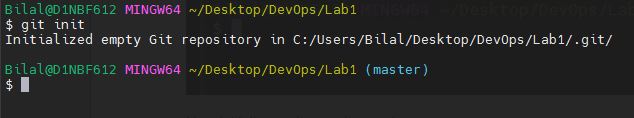
</html>

Press **:wq** and press enter. Now your folder should look like this:



Now initialize a git repository using:

git init

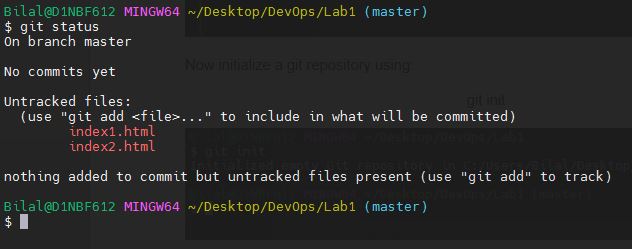


The terminal shows that we are now in the master branch of the newly created local Git repository.

## Check Status

To check what changes are being tracked by Git, run:

git status

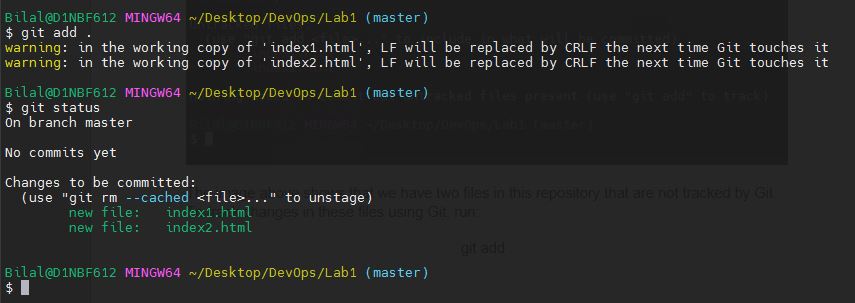


The image above shows that we have two files in this repository that are not tracked by Git.

## Add Files to Staging Area

To track changes in these files using Git, run:

git add .



The image above shows that the files are now being tracked by Git but are not commited yet.

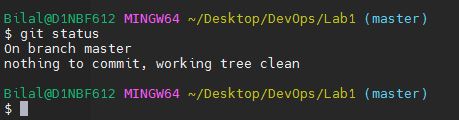
## Commit Changes

To commit the changes to Git, run:

git commit –m “[message here]”



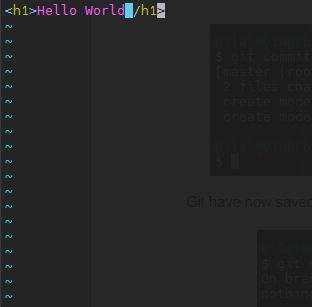
Git have now saved a snapshot of our repository. Check status of repository:



Now we change index1.html to:

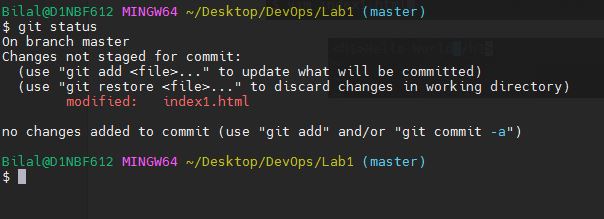
<h1>Hello World</h1>

C:\Users\Bilal\Desktop\1.JPG



Now check status. Run:

git status

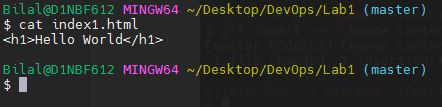


The output shows that one file “index1.html” has been modified and has not been committed to Git. We now add and commit the changes. Run:

git add .

git commit –m “Change index1.html contents to just Hello World”

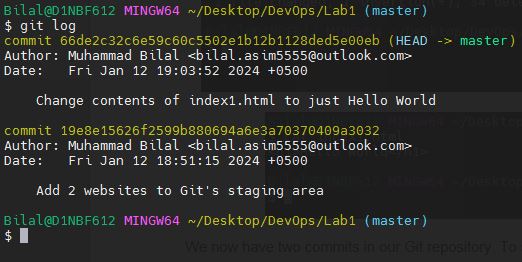




## View History

We now have two commits in our Git repository. To see commit history, use command:

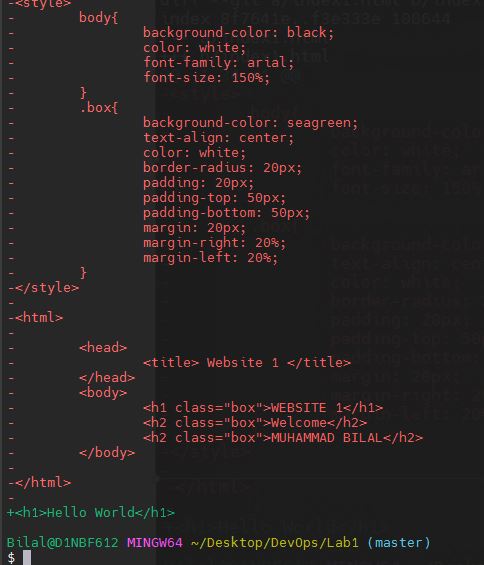
git log



We now have two snapshots of our repository. To view contents of a specific commit, run:

git show [HEAD~Offset / commit ID]

HEAD is a pointer that points to latest commit. Offset tells Git to go back how many commits. Alternatively, we can just write unique commmit ID of a specific commit.

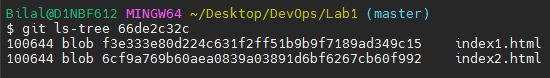


Git show command shows what changes were recorded in this commit. Lines that were removeed in this commit are shown with a minus sign ( - ) before them. Lines with a plus sign ( + ) show that they were newly added in that commit.

## View Contents of Older Commit

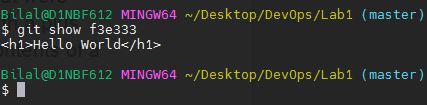
To view, complete contents of a specific commit instead of changes. Run:

git ls-tree [commit unique id]



In the output above, blob represents file and tree represents directory. To view contents of a specific file in this commit, we run the git show command with unique id of that file from the image above:

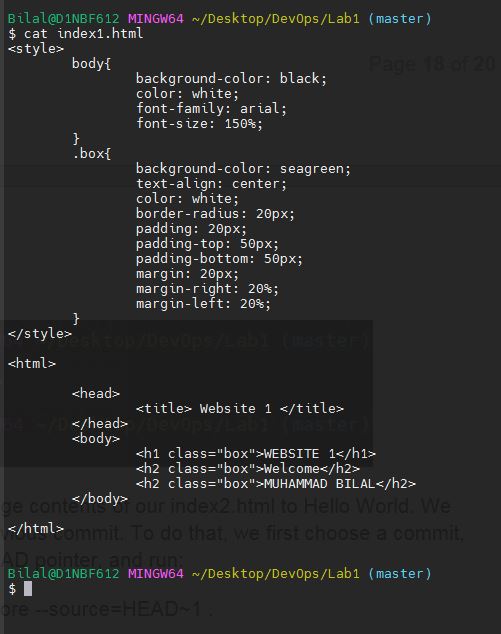
git show [file’s unique ID]



## Restore Previous Commit

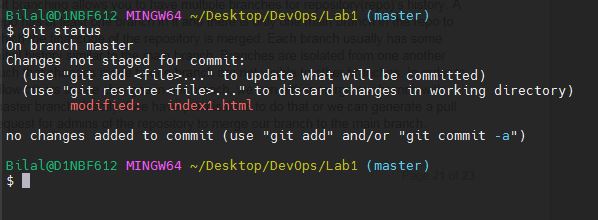
Now suppose we did not want to change contents of our index1.html to Hello World. We need to roll back our changes to a previous commit. To do that, we first choose a commit, copy its unique id or its offset from HEAD pointer, and run:

git restore --source=HEAD~1 .



We can see in the above image that we have successfully restored index1.html to our original site.

Now, first we need to make another commit to save our changes to Git.



Run:

git add .

git commit –m “restored to a previous commit”



## Git Branching

Git branching allows you to have multiple branches for repository(repo)’s history. A repo has at least one branch in it and there is only one main branch in the repo to which the final code of the repository is merged. Each branch usually has some initial history similar to the main branch. Branches are isolated from one another such that changes made to one branch are not visible to other branches. In order to allow others to see changes in our branch, we can either merge our branch with master branch directly if we have permissions to do that or we can generate a pull request for admins of the repository to merge our branch to the main branch.

In order to Create a new branch, run:

git checkout –b “new-branch1”



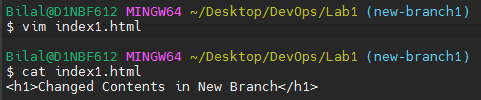
-b flag tell git to create a new branch. In order to switch to an existing branch, we don’t need the –b flag. To view branches in your current repository and your current branch, run:

git branch



We now change index1.html in our new branch to:

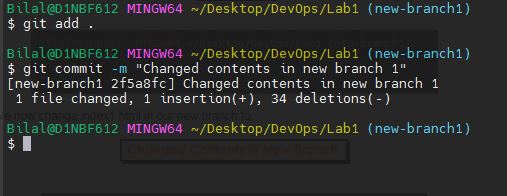
Changed Contents in New Branch



Commit changes:

git add .

git commit –m “Changed contents in new branch 1”

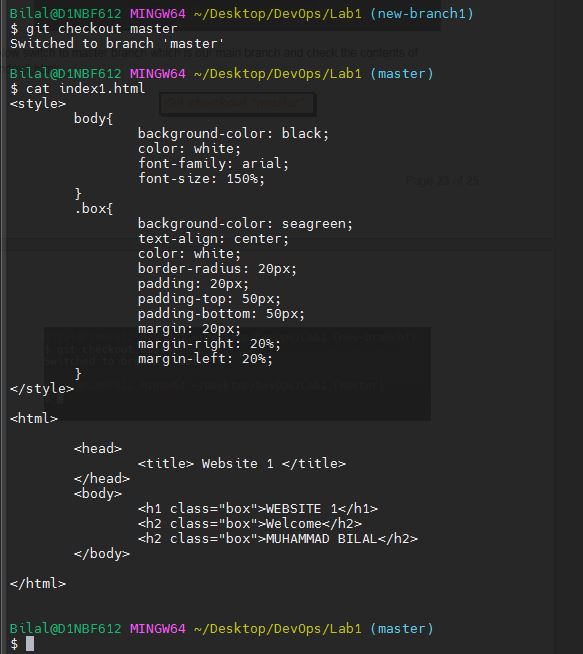


Now switch to master branch which is our main branch and check the contents of index1.html:

Git checkout “master”



cat index1.html

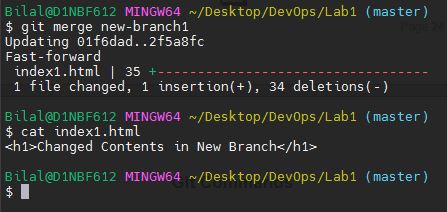


We can see in the above output that the contents of index1.html have not changed. In order to reflect changes into the master branch, we need to merge the new-branch1 and master branch. To do that, run:

git merge new-branch1

cat index1.html

# 



# Git Commands

· Create git repository and upload to github

· Practise basic git commands

· Git branching

· Git config

· Create Document on Git commands